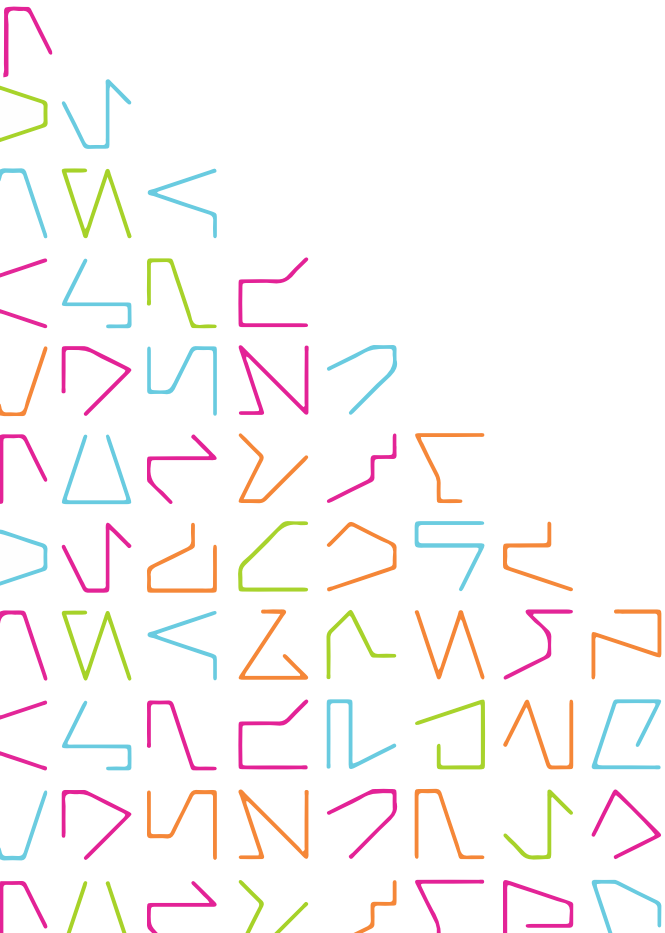
TAMPEREEN
AMMATTIKORKEAKOULU

TESTAUKSEN AUTOMATISOITI JA MAHDOLLISUUKSIA TIETOLIIKENNEJÄRJESTELMÄSSÄ

Juhani Nieminen



Opinnäytetyö
Toukokuu 2016
Tietotekniikka
Tietoliikennetekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan ko.
Tietoliikennetekniikka

NIEMINEN JUHANI:
Testauksen automatisointi ja mahdollisuuksia tietoliikennejärjestelmässä

Opinnäytetyö 31 sivua, joista liitteitä 0 sivua
Toukokuu 2016

Tässä työssä käsitellään testauksen automatisoinnin perusperiaatteita, hyötyjä, riskejä ja mahdollisuuksia tietoliikennejärjestelmässä. Työn tarkoitus on luoda pohja reitittimen testausautomaation jatkokehitykselle.

Ohjelmistokehityksessä tehokkuuden lisääminen on tullut tärkeäksi. Ohjelmistoja ja laitteita halutaan entistä halvemmalla ja nopeammin valmiiksi. Yksi vaihtoehto ohjelmistokehityksen tehostamiselle on automatisoida testaus niin pitkälle kuin mahdollista. Testaus syö suuren osan koko ohjelmistokehityksen ajasta, koska kaiken halutaan toimivan ennen asiakkaalle toimittamista. Tämä aiheuttaa sen, että tullaan toistamaan samoja testejä useaan kertaan koko ohjelmistokehityksen aikana. Hyvin toteutettu testauksen automatisointi säästää aikaa ja mahdollistaa laadukkaampien tuotteiden synnyn.

Testauksen automatisointi noudattaa samoja perusperiaatteita kuin muukin testaus. Tavoitteena on toteuttaa automaattisesti samat testit, jotka muuten tehtäisiin käsin. Reitittimellä tämä tarkoittaa esimerkiksi laiteen asetuksien tarkistamista. Jotta se olisi mahdollista, täytyy ensin kirjautua laitteelle ja antaa komento, joka tulostaa reitittimen sen hetkiset asetukset näytölle. Tämän jälkeen vertaillaan reitittimellä olevia asetuksia haluttujen asetuksien kanssa. Vaikka asetukset olisivat oikein, ei voida varmasti todeta, että reititin toimisi niiden mukaan. Tämän vuoksi on tarkistettava jollain muulla tavalla, että reititin toimii asetuksien mukaan. Reitittimen yksi tehtävä on reititys, joka mahdollistaa eri verkkojen välisen liikenteen. Reitityksen toimivuus on helppo testata esimerkiksi lähettämällä viesti verkkojen välillä. Reitittimellä on myös muita tehtäviä, jotka on hyvä ottaa huomioon testauksessa.

Testauksen automatisointiin tarvitaan työkalu tai työkaluja, joilla voidaan toteuttaa varsinainen automatisointi. Näitä työkaluja on olemassa moneen eri käyttötarkoitukseen. Yksiselitteisesti parasta ei ole, vaan sopivin työkalu riippuu siitä, mitä sillä halutaan saavuttaa. Tietoliikennejärjestelmässä se voisi olla esimerkiksi viestien lähettämistä verkosta toiseen automaattisesti. Jos valmiit työkalut eivät toteuta haluttuja asioita, niitä voidaan muokata ja kehittää omiin tarpeisiin sopiviksi.

Reitittimen testauksen automatisointiin on olemassa erilaisia työkaluja. Jotkut työkalut on kehitetty vain yhden reitittimen testaukseen ilman muita verkkokomponentteja. Jotkut voivat toteuttaa koko verkkotestauksen nauhoittamalla annettuja käskyjä, kun taas toiset on kehitetty testausautomaatioalustoiksi.

Asiasanat: Testauksen automatisointi, verkko, reititin

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information Technology
Telecommunications

NIEMINEN JUHANI:
Automated Testing and its Possibilities in Network Testing

Bachelor's thesis 31 pages, appendices 0 pages
May 2016

This thesis handles test automation and its basic principles, benefits, risks and opportunities in a digital communication system. The purpose of this study is to create a basis for test automation for further development.

Increasing efficiency has become an important part of software development. Software and hardware should be even cheaper and faster to complete. One option to step up the development is to automate the testing of the product. Testing takes an enormous part of the overall time spent on software development since all the features must work before delivery. Therefore the same test will be repeated several times throughout the life cycle of the software development. Well implemented test automation saves lots of time and allows better-quality products to be created.

The first two parts introduce a software-based testing, network technology, as well as the function of the router. The next section examines the automation of testing and its principles. Finally, thesis will address network testing methods and what kinds of tools exist to automate network testing.

Test automation follows the same basic principle as all other testing. You can make automated tests as you would do them manually. When testing router you'd like to confirm that settings are as they are supposed to be. But that is not enough, since a device can work differently than it says. Therefore there must be some kind of an approach to test if the router really works. You can try and send messages from one network to another to verify that the routing works throughout.

In order for the test automation to be successful a tool of some kind is needed. There are already many types of tools for different kinds of use. Some are made for automating everything and others are just for one kind of use. There are different kinds of tools to automate testing for routers. One tool is easy to use and verifies that the router works as the protocol assumes. Others are more versatile and can be used to test the entire network.

Key words: Test automation, network, router

SISÄLLYS

LYHENTEET JA TERMIT

ARP	Address Resolution Protocol, tarkistaa IP-osoitetta vastaavan MAC-osoitteen
MAC-address	Media access control address, fyysinen osoite
TCP	Transmission Control Protocol, yhteydellinen tiedonsiirtoprotokolla
IP	Internet Protocol, IP-protokolla
UDP	User Datagram Protocol, yhteydetön tiedonsiirtoprotokolla
ICMP	Internet Control Message Protocol, nopea viestinvälittäjäprotokolla
IGMP	Internet Group Management Protocol, mahdollistaa multicast-ryhmään liittymisen
LAN	Local Area Network, lähiverkko
WAN	Wide Area Network, laaja verkko
BGP	Border Gateway Protocol, BGP-reititysprotokolla
OSPF	Open Short Path First, OSPF-reititysprotokolla
NAT	Network address translation, osoitteenmuunnosprotokolla
DHCP	Dynamic Host Configuration Protocol, protokolla, jota käytetään IP-osoitteen, nimipalvelimen ja oletusyhdyskäytävän jakamiseen verkkoon kytketyille verkkolaitteille
VPN	Virtual private Network, virtuaalinen erillisverkko
IETF	Internet Engineering Task Force, internetprotokollien standardisointiorganisaatio
SNMP	Simple Network Management Protocol, IP-verkkojen hallintaan käytettävä tietoliikenneprotokolla
VLAN	Virtual LAN, Virtuaalinen lähiverkko
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla
SSH	Secure Shell, salattu tietoliikenneprotokolla
IKE	Internet Key Exchange, salausavainten vaihtoprotokolla
ESP	Encapsulating Security Payload, pakettisalausprotokolla

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on tutkia mahdollisuuksia toteuttaa testausautomaatiota tietoliikennejärjestelmässä etenkin reitittimen osalta. Lisäksi käsitellään, mitä on testauksen automatisointi. Testausautomaation tavoitteena on helpottaa ohjelmistokehityksen prosessia, säästää aikaa ja rahaa sekä antaa nopeaa palautetta koko ohjelmistokehityksen ajan. Tietoliikennejärjestelmässä merkittävin yksittäinen verkkolaite on reititin, jonka toiminta halutaan varmistaa automaattisesti. Tietoliikenneverkko tuo oma lisänsä testauksen automatisoinnin toteutukseen.

Aluksi käsitellään ohjelmistotestausta ja tietoliikennetekniikkaa, minkä jälkeen esitellään testauksen automatisoinnin hyödyt ja haitat. Lopuksi tutkitaan, mitä pitää huomioida reititintä testattaessa ja mitä työkaluja on olemassa tietoliikennejärjestelmän automaattiseen testaukseen. Varsinaisia testaustyökaluja, joita tarkastellaan, on valittu kolme. Työn tarkoituksena on toimia reitittimen testauksen automatisoinnin suunnittelun pohjana ja perustella, mitä hyötyä testauksen automatisoinnista on.

2 OHJELMISTOTESTAUKSESTA

Ohjelmistotestaus on ohjelmistotuotannon osa-alue. Ohjelmistotestauksella varmistetaan tuotettavan ohjelmiston toimiminen ja suunnitelmien mukainen toteutus. Testauksen tehtävänä on varmistaa, että valmistetaan oikeaa tuotetta ja että tuote on tehty oikein asiakasvaatimusten mukaan. (Kasurinen 2014, 9-10)

Testaus on laaja käsite ja kokonaisuutena laajempi kuin esimerkiksi pelkkä ohjelmointi tai tekninen kirjoittaminen. Testaajalta vaaditaan paljon osaamista ohjelmistokehityksessä ja työnantajasta riippuen voi joutua tekemään useaa erityyppistä tehtävää työvaiheista riippuen. Tärkeää testauksessa on tietää ohjelman käyttötarkoitus. Joillekuille voi olla tärkeää graafisten elementtien toimivuus, kun taas toisille toiminallisuus on merkityksellisintä. (Kasurinen 2014, 12–15)

Usein testaus saatetaan jättää vähälle huomiolle, koska sitä ei nähdä suoraan tuottavana toimintana yritykselle. Esimerkiksi vesiputous-ohjelmistokehitysmallissa testaamisen jälkeen jokin ominaisuus voi palata takaisin kehittäjälle, joka korjaa mahdollisen virheen ja näin käyttää toistamiseen työaikaansa samaan tehtävään. Tutkimusten mukaan yhdysvaltalaiset yritykset menettivät vuonna 2002 noin 21 miljardia dollaria puutteellisen testauksen johdosta tuotteisiin jääneiden virheiden aiheuttamina tappioina. Kun lisätään tähän asiakkaille aiheutuneet tappiot ja tuotantomenetykset, todelliset menetykset olivat noin 60 miljardia dollaria. Ohjelmistokehityksessä lopputestaus on viimeinen aikaa vievä vaihe. Tämän vuoksi testausta saatetaan syyttää ohjelmiston myöhästymisestä. (Tassey 2002)

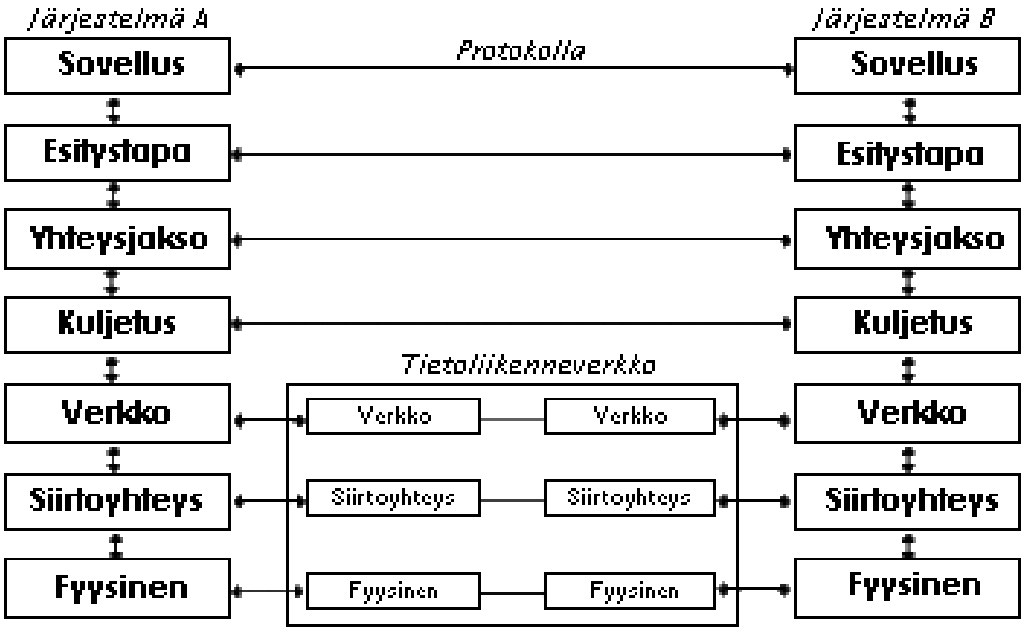
Lopputestaus on ohjelmistotuotannon viimeinen vaihe, jonka toteutusaikataulu suunnitellaan usein yltiöoptimistiseksi. Sen takia voidaan joutua toimittamaan heikosti testattuja ohjelmia, joiden toiminnasta ei voida olla varmoja. Kehityksen aikana testaus voi olla uuden ominaisuuden nopeaa testausta, jolloin varmistetaan vain sen toiminta. Näin testattaessa voi jäädä kokeilematta muita ominaisuuksiin liittyviä asioita, jotka ovat voineet uutta ominaisuutta toteutettaessa jäädä huomioimatta. (Kasurinen 2014, 16–19)

Ohjelmistotestaustyö kokonaisuutena on enemmän kuin vain ohjelman koeajamista ja ohjelmavirheiden dokumentointia. Työhön kuuluvat oleellisesti työkalut, menetelmät, testien suunnittelu, laatuvaatimukset, kehitystyön vaatimukset sekä henkilöstö. (Kasurinen 2014)

3 VERKKOTEKNIIKKA

3.1 Verkkoprotokollat

Verkkoprotokollat määräävät, millä tavalla ja miten data liikkuu verkossa. Protokollia on useita, jotka jaetaan useisiin eri kerroksiin, joilla on oma tehtävänsä ja tarkoituksensa. Kerrosten jaottelussa on kaksi määräävää mallia, jotka ovat OSI- ja TCP/IP-malli. OSI on kansainvälisen standardointijärjestö ISO:n kehittämä malli internetverkon vaatimuksista (Colliander 1999). TCP/IP on maailmalla enemmän käytössä, ja se on IETF-organisaation standardoima (Parziale ym. 2006, 29, 47). Nämä mallit eroavat kerroksissaan ja niiden määrissä. OSI-mallissa on seitsemän kerrosta, jotka ovat fyysinen, siirto-, verkko-, kuljetus-, istunto- tai yhteysjakso-, esitystapa- ja sovelluskerros (Colliander 1999). TCP/IP-mallissa on vain neljä kerrosta, joista osaan sisältyy OSI-mallin kerroksia. Näiden kerrosten nimet ovat fyysinen, verkko-, kuljetus- ja sovelluskerros (Parziale ym. 2006, 33). Vaikka TCP/IP-malli on yleisesti enemmän käytössä oleva malli, tässä työssä tutkitaan OSI-mallia, sillä sen protokollakerrokset ovat helpommin jaettavissa omiin tarkoituksiinsa (kuva 1).



KUVA 1. OSI-malli (http://www.cs.tut.fi/etaopetus/titepk/kuvat/OSI1.gif)

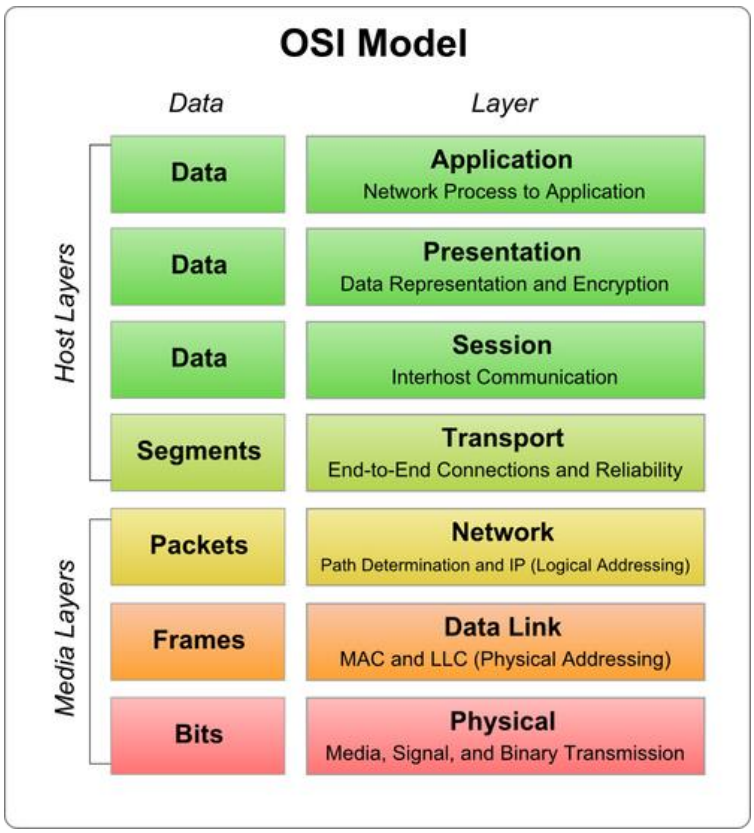
Ylimpänä on sovelluskerros, jonka tehtävänä on toimia rajapintana sovelluksille, jotka käyttävät verkkoa. Esitystapakerros pitää huolta, että välitettävä tieto esitetään sovellus-

tai istuntokerrokselle oikeassa muodossa. Se ei ota kantaa siihen, miten tietoa siirretään. Viides kerros on yhteysjakso- eli istuntokerros. Sen tarkoituksena on luoda yhteys kahden tai useamman istunnon välille, joten se aloittaa tai lopettaa tiedonsiirtoon tarvittavan yhteyden. Neljäntenä on kuljetuskerros. Sen tehtävänä on huolehtia tiedonsiirron toteutuksesta. Se pilkkoo tiedon tarvittaessa pienempiin osiin ja varmistaa, että tieto saapuu vastaanottajalle. Kuljetuskerroksen tehtävänä on myös muodostaa erilaisia yhteyksiä, kuten UDP tai TCP. (Colliander 1999)

Verkko-, siirto- ja fyysinen kerros OSI-mallissa vastaavat TCP/IP-mallin kahta alinta eli verkko- ja fyysistä kerrosta. Kuvassa 1 nämä on merkitty omaan laatikkoonsa, jonka nimi on tietoliikenneverkko. Todellisuudessa nämä viimeiset kerrokset vastaavat datan kuljettamisesta verkossa oikeaan kohteeseen. Kerrokset eivät takaa yhteyden luotettavuutta tai korjaa verkkoyhteyksien virheitä. Nämä ominaisuudet hoidetaan ylemmillä kerroksilla. (Parziale ym. 2006, 34)

Verkkokerros eli internet-protokolla toteuttaa verkon reitityksen ja pakettien toimituksen oikeisiin osoitteisiin. Sen tärkein tehtävä on määrittää, kuinka eri verkot voidaan yhdistää. Protokolliin kuluu esimerkiksi IGMP- ja ICMP-protokollat. Kerros käsittelee myös IP-osoitteita, joita on käytössä kahdenlaisia: IPv4- ja IPv6-osoitteita. Verkkokerroksella määritellään myös, miten liikennettä välitetään eteenpäin, onko kyseessä täsmälähetys eli unicast, yleislähetys eli broadcast, ryhmälähetys eli multicast tai joukkolähetys eli anycast. (Parziale ym. 2006, 103, 109)

Kuvassa 2 on esitetty toinen versio OSI-mallista ja siitä, mihin kerrokset liittyvät. Fyysisellä tasolla kerrotaan siis, millä medioilla tieto siirretään. Siirtokerroksella määritellään fyysiset osoitteet eli laitteiden MAC-osoitteet, jotka ovat yksilöllisiä jokaiselle ARP-kyselyissä käytettävälle laitteelle. Kerros määrittää myös internetpakettien enimmäiskoon, jotta niitä pystytään lähettämään tehokkaasti. Yksi tunnetuimmista protokollista on Ethernet. Siirtokerros siis määrittelee, minkälaisia ja -kokoisia paketteja verkossa voidaan siirtää. Siirtokerros rajaa myös todellisen enimmäistiedonsiirtonopeuden, vaikka sitä ei tällä tasolla määritellä. (Parziale ym. 2006, 55)



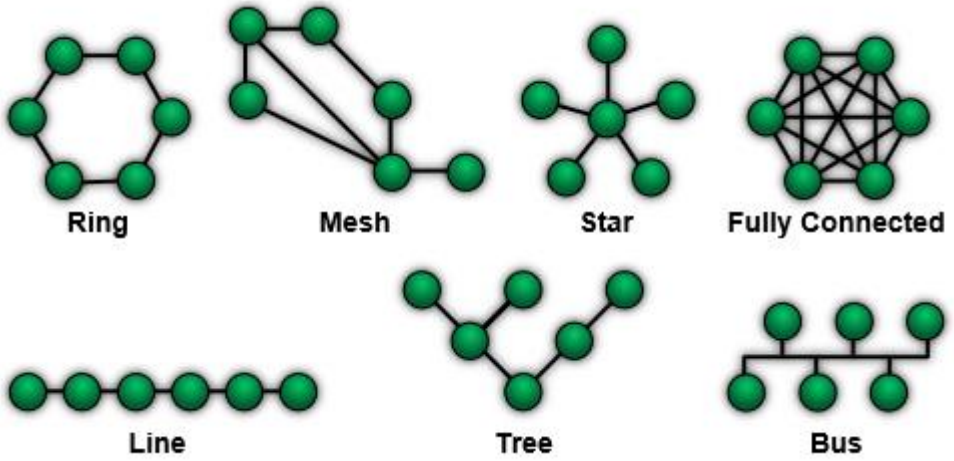
KUVA 2. OSI-malli (http://www.tech-faq.com/wp-content/uploads/2009/01/osimodel.png)

3.2 Tietoverkkojen rakenne ja topologiat

Tietoverkot rakentuvat sisä- ja ulkoverkoista eli LAN- ja WAN-verkoista. Sisäverkolla tarkoitetaan maantieteellisesti suhteellisen pientä aluetta, kuten koulua tai yritysten toimitiloja, ja siihen kuuluu usein erilaisia laitteita, kuten työpisteitä, printtereitä ja palvelimia. WAN- eli ulkoverkolla taas tarkoitetaan maantieteellisesti suurempaa aluetta, kuten Suomea, Eurooppaa tai koko maailmaa. Sisäverkot ovat yhteydessä ulkoverkkoihin, mikä mahdollistaa tiedon välittämisen maapallon toiselta puolelta toiselle silmänräpäyksessä. Tämä on mahdollista verkossa olevilla laitteilla, jotka ohjaavat ja välittävät tietoa. Tällaisia laitteita ovat kytkimet ja reitittimet. (Parziale ym. 2006, 122)

Verkkotopologiat kuvaavat verkon loogista rakennetta eli miten saman verkon laitteet kytkeytyvät toisiinsa. Topologia voi kuvata myös fyysistä rakennetta, mutta tällöin täytyy ottaa huomioon rakennuksen mahdolliset kerrokset ja kaapelien pituudet. On siis miele-

kästä kuvata verkot omina loogisina topologioinaan. Perustopologiakuvauksia on seitsemän, jotka on esitetty kuvassa 3. Topologiat voivat myös olla kaikkien näiden yhdistelmiä. (Parziale ym. 2006, 125–126)



KUVA 3. Verkkotopologiat (https://en.wikipedia.org/wiki/Network_topology)

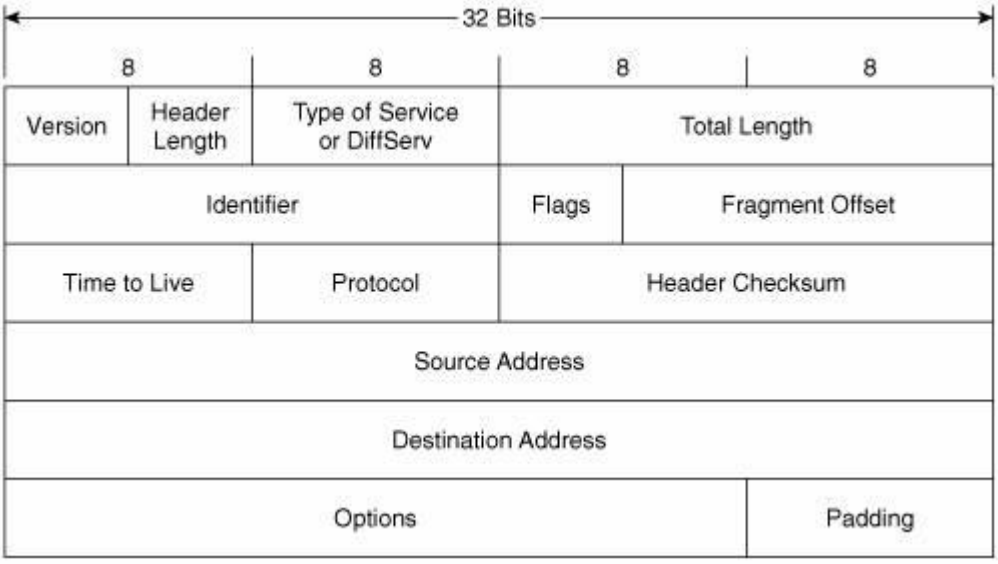
Line-topologia on yksinkertaisin ja helpoin toteuttaa, joka ei ole enää suuressa käytössä virhe herkkyyden ja tietoturvan takia. Bus-topologiassa kaikki päätelaitteet ovat yhdistettynä yhteen kaapeliin. Se mahdollistaa halvan asennuksen, mutta voi myös aiheuttaa ongelmia; kaikki data, jota kaapelissa lähetetään käy nimittäin kaikilla koneilla löytääkseen oikean vastaanottajan. Kuka tahansa, joko pääsee kytkeytymään pääkaapeliin voi kuunnella siellä kulkevaa tietoa. Toinen mahdollinen, suurempi ongelma on pääkaapelin rikkoutuminen, mikä aiheuttaisi verkkoyhteyksien katkeamisen kaikilta.

Ring-topologia on vanhanaikainen malli, jota ei nykyään käytetä. Sitä on käytetty lähinnä silloin, kun signaali on heikentynyt matkalla ja sitä on haluttu vahvistaa. Star-topologiassa jokainen kone on kytketty hubiin, joka vahvistaa ja toistaa signaalia. Nykyään hubin tilalla voisi olla esimerkiksi kytkin, jolla saa hubin verrattuna lisää mahdollisuuksia verkkoliikenteen hallintaan.

Mesh-topologiassa kaikki toimipisteet ovat yhteydessä toisiinsa, mikä on hyödyllistä silloin, kun halutaan luotettavuutta ja varmuutta verkkoon. Jos kahden toimipisteen välinen yhteys katkeaisi, liikenne voidaan ohjata kolmannen toimipisteen kautta, jolla on yhteys näihin kahteen toimipisteeseen. Tämä on kalliimpi kuin muut topologiat, mutta hyödyllinen etenkin toimintavarmuuden kannalta. Tree-topologiassa on yksi pääpiste, josta

verkko haaroittuu useampaan polkuun. Tällainen malli mahdollistaa esimerkiksi yksittäisten alueiden itsenäisen toimimisen ilman, että liikenteen tarvitsisi siirtyä muille osapuolille. (Parziale ym. 2006, 126–131)

Tietoliikenneverkossa kulkeva tieto paketoidaan standardien mukaisesti, jotta se kulkisi mahdollisimman tehokkaasti ja luotettavasti lähettäjältä vastaanottajalle. Tieto pilkotaan pienempiin paketteihin, ja jokaiseen pilkottuun pakettiin on lisättävä otsikkokenttä, joka kertoo, onko paketti IPv4- vai IPv6-paketti, minkä kokoinen se on, lähettäjän ja vastaanottajan IP-osoitteen sekä paljon muuta hyödyllistä informaatiota. Kuvassa 4 on IPv4-version paketin otsikkokenttä. Tähän otsikkokenttään tulee tieto myös siitä, mitä protokollia käytetään paketin lähetyksessä. Jos paketti on esimerkiksi OSPF-viesti toiselle reitittimelle, protocol-kenttään tulee arvo 89, joka siis kertoo paketin olevan OSPF-viesti. (Doyl, Carrol 2005)



KUVA 4. IPv4-paketin otsikkokenttä (http://flylib.com/books/en/2.296.1.21/1/)

3.3 Reititin ja sen tehtävät

Yksi tärkeimmistä verkon komponenteista on reititin. Se kuljettaa paketteja lähettäjältä vastaanottajalle (Parziale ym. 2006, 136). Verkkolaitteissa sillä on suuri merkitys verkon toiminalle. Reititin voi muokata pakettien muotoa, analysoida verkkoliikennettä ja välittää paketteja eri verkkojen välillä (Router, 2016).

Reititys tapahtuu OSI-mallin kolmannella kerroksella eli verkkokerroksella. Sen tehtäviä ovat optimaalisen reitin löytäminen lähettäjältä vastaanottajalle ja pakettien kuljettaminen verkosta toiseen. Ilman reititystä ja reititintä olisi hankalaa yhdistää kaksi sijainniltaan erillään olevaa verkkoa. Vaikka reitittimen päätehtävänä on pakettien siirtäminen lähettäjältä vastaanottajalle, sillä voi olla myös monia muita tehtäviä, verkkoliikenteen salaus ja DHCP-palvelu. (Parziale ym. 2006, 136)

Verkot jaetaan omiksi kokonaisuuksiksi aliverkottamalla, jolloin ilman reititystä vain samassa aliverkossa olevat laitteet voivat välittää paketteja keskenään. Verkkoja voidaan jakaa myös käyttäen virtuaaliverkkoja eli VLAN-tekniikkaa. Tällöin ainoastaan samassa virtuaaliverkossa olevat laitteet ovat yhteydessä toisiinsa riippumatta aliverkotuksesta. Reititin voi kuitenkin reitittää liikennettä eri VLAN:ien välillä. (Configuring VLANs, 2016)

Reititysprotokollien tehtävänä on määrittää pakettien reitti ja siten yhdistää eri verkot toisiinsa. Ne pystyvät algoritmiensa avulla ohjaamaan verkkoliikenteen automaattisesti oikeaan paikkaan ilman, että käyttäjän tarvitsee huolehtia jokaisesta reitistä erikseen. Reititys voi olla kahdenlaista, joko staattista tai dynaamista. Kaksi tunnetuinta käytössä olevaa dynaamista reititysprotokollaa ovat OSPF ja BGP. OSPF-reititystä käytetään usein sisäverkoissa. BGP on käytössä isommissa runkoverkoissa. Joissakin tapauksissa on kuitenkin mielekkäämpää käyttää staattista reititystä, jolla voidaan toteuttaa pienemmän verkon reititystarpeet. (Parziale ym. 2006, 200–201)

DHCP-palvelu on yksi reitittimen mahdollisista tehtävistä reitityksen lisäksi. DHCP-palvelu voi olla myös erillisellä palvelimella. Sen tarkoituksena on jakaa verkossa vapaita IP-osoitteita niitä tarvitseville laitteille. Langattomissa verkoissa käytetään usein DHCP-palvelu helpottamaan uusien laitteiden kytkemistä. Se voi toimia kahdella tavalla: joko se vuokraa osoitteita, jolloin laitteen IP-osoite uusitaan esimerkiksi päivittäin, tai se jakaa pysyviä IP-osoitteita. Tällöin verkkoon kytketyn laitteen ei aina tarvitse hakea uutta osoitetta ja laitteiden hallinnointi on helpompaa. (Configuring DHCP, 2006)

Kaikilla internetiin yhdistetyillä laitteilla tulee olla yksilöllinen IP-osoite. IPv4-osoitteet ovat jo loppuneet, joten kaikille laitteille ei voida antaa omaa IP-osoitettaan. NAT-palvelun tehtävänä on mahdollistaa pääsy ulkoverkkoon sisäverkosta, jossa on monta käyttäjää

samalla IP-osoitteella. Kaikki julkisessa internetissä olevat IPv4-osoitteet ovat rekisteröityjä, ja koska niitä on vain rajattu määrä, niin niitä ei riitä kaikille internetiin yhdistetyille laitteille. Siksi on kehitetty NAT, joka mahdollistaa kaikkien laitteiden pääsyn julkiseen verkkoon vain yhdellä IP-osoitteella. (Configuring Network Address Translation: Getting Started, 2014)

VPN on virtuaalinen erillisverkko, jolla voidaan yhdistää kaksi tai useampi erillään olevaa verkkoa ja salata verkossa kulkevaa liikennettä. VPN:n avulla voidaan myös yhdistää yksittäisiä koneita tai käyttäjiä muualla sijaitsevaan sisäverkkoon. VPN on siis tunnelointitekniikka, jolla voidaan yhdistää kaksi yksityisverkkoa internetillä. VPN ei itse salaa tunnelissa kulkevaa liikennettä, mutta yleensä VPN yhdistetään salattuun tietoliikenteeseen. Tämä toteutetaan IETF:n standardoimalla IPsec-salauksella, jolla todennetaan ja salataan yhteys. OSI-mallissa IPsec toimii verkkokerroksella. IPsec:in avulla voidaan myös tarkistaa, ettei saatua dataa ole muokattu ja uudelleenlähetetty vastaanottajalle. IPsec:illä salatut paketit näkyvät verkossa ESP-paketteina. ESP on kapselointitekniikka, jolla salataan paketit ja estetään niiden uudelleen lähettäminen. IKE-protokolla yhdistetään aina VPN-salaukseen. Se on avainten jako- ja vaihtotekniikka, jolla voidaan varmistaa ja todentaa yhteyden muodostuminen sallittujen osapuolien välille. (Configuring IP-Sec Network Security, 2016)

Verkon virhetilanteita varten reitittimet voivat kirjata huomaamiaan asioita erilliselle palvelimelle. Kirjaus voi kertoa esimerkiksi pakettien häviämisestä tai hylkäämisestä, reitittimen virhetilanteista, verkkoon liittyneistä laitteista tai reitittimen asetusten muuttumisesta. Palvelimelta voidaan siten tutkia verkon toimintaa ilman, että käyttäjä tarvitsee yhteyttä reitittimille. (Troubleshooting, Fault Management, and Logging, 2001)

Reititin voi toimia myös joiltakin osin palomuurina, jonka tarkoituksena on rajata verkossa kulkevaa liikennettä. Se voi tarkoittaa esimerkiksi pääsyn estämistä internet-verkossa oleville verkkosivuille tai verkossa toimivien protokollien rajoittamista. Suurissa verkoissa palomuuri on usein erillinen laitteensa, mutta pienissä verkoissa reitittimen avulla toteutettu palomuuri on jo hyvällä tasolla. (Configuring a Simple Firewall, 2016)

4 TESTAUKSEN AUTOMATISOINNISTA

4.1 Testauksen perusperiaatteita

Automaattinen testaus voi tarkoittaa eri asioita eri konteksteissa. Kehittäjä voi mieltää sen yksikkötestien tekemisenä, kun taas testaajalle se voi tarkoittaa käyttöliittymätestien nauhoittamista ja nauhoitusten toistamista apuohjelman avulla. Kysymys on lähinnä siitä, mitä halutaan automatisoida. Automaattiseen testaukseen liittyy rasitus-, luotettavuus-, turvallisuus- ja suorituskykytestaus. Testausautomaatio voi olla kaikkea tätä, mutta mielikuvat vaihtelevat. Tässä luvussa tutustutaan siihen, mitä testausautomaatio on, miksi se on kannattavaa ja mitä sen toteuttaminen vaatii. (Dustin, Garret & Gauf 2009, 34)

Automaattiselle testaukselle ei ole selkeää toimintamallia. Toteutettava automaattinen testaus riippuu ympäristöstä, tuotteesta ja teknologiasta, mutta ei esimerkiksi ohjelmistotuotantomallista. Usein kaikki käsin tehtävä testaus on mahdollista automatisoida, vaikka se ei olisikaan taloudellisesti kannattavaa. Testauksen automatisoinnin onnistuminen ei tarkoita välttämättä kaikkien testien automatisointia, vaikka monet sitä varmasti tavoittelevatkin. (Dustin ym. 2009, 35)

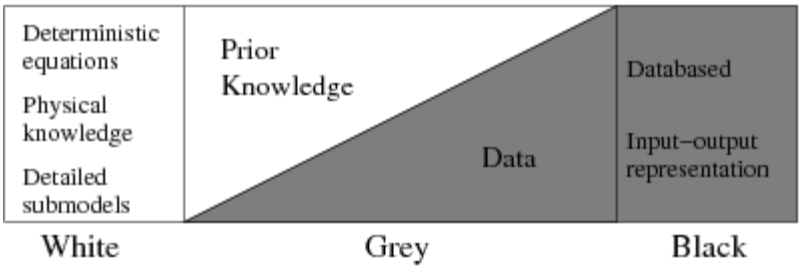
Malaiyan (1999) mukaan ohjelmistotuotanto on jäänyt jälkeen muista tuotantoaloista, etenkin tuotteen testauksen osalta. Esimerkiksi autot ja tuotantokoneet suunnitellaan simulaattorien ja ohjelmistojen avulla, jolloin säästetään aikaa ja rahaa verrattuna samaan prosessiin ilman suunnitteluohjelmistoja. Ohjelmistokehityksessä ja ohjelmistojen testauksessa tällaisia suunnitteluohjelmia ei ole, vaikka ohjelmistot ovat huomattavan monimutkaisia ja ne hyötyisivätkin testauksen automatisoinnista.

Automaattinen testaus käsin testaamiseen verrattuna voi painottua sellaisiin testeihin, joita on vaikeaa tai työlästä toteuttaa käsin, kuten ohjelmistojen mahdollisten muistivuotojen pitkäaikaiset testit. Testauksen automatisointi on käsin testaamista lähempänä ohjelmistokehitystä, mutta silti testaajalla on tärkeä rooli tulosten analysoinnissa. Lisäksi testaajan täytyy ymmärtää, kuinka testaus kuuluisi toteuttaa, jotta tulokset olisivat luotettavia. Testauksen automatisointi ei kuitenkaan useimmissa tapauksissa tule poistamaan manuaalisen testauksen tarvetta ohjelmistotuotannossa. (Dustin ym. 2009, 36)

Testauksen automatisoinnin mahdollisimman tehokas toteutus vaatii tietoa siitä, mitä ja minkälaisia testejä automaattisessa testauksessa tarvitaan. Automaattisen testauksen on toimittava usealla laitteella tai tietokoneella yhtäaikaisesti, mikäli testattava tuote sitä vaatii. Parhaimmillaan hyvä automaattinen testaustyökalu toimii eri ohjelmointikielillä toteutetuissa ohjelmistoissa, jolloin esimerkiksi riippuvuus käyttöjärjestelmästä poistuu. Automaattisen testauksen olisi hyvä toimia sekä graafisella että tekstipohjaisella käyttöliittymällä. Olisi hyvä ottaa huomioon myös mahdolliset testattavat rajapinnat, jotta niitä voisi testata samalla automatisoidulla testausohjelmistolla.
(Dustin ym. 2009, 36)

Automaattinen testaus voi koostua useasta yhtä aikaa toimivasta työkalusta. Näiden tulisi toimia yhdessä riippumatta siitä, ovatko ne vapaaseen lähdekoodiin perustuvia, kaupallisia tuotteita vai itsekehitettyjä. Jos testauksen tueksi on jo kehitetty joitakin testaustyökaluja, niitä kannattaa hyödyntää. Automatisoinnista saa parhaan avun silloin, kun sitä käytetään ja kehitetään ohjelmiston rinnalla koko tuotteen elinkaaren ajan. Parhaimmillaan automatisointi on yksinkertaisesti toteutettua, jolloin sitä voidaan käyttää uusissa tuotteissa ja ympäristöissä pienillä muutoksilla. (Dustin ym. 2009, 38)

Automatisoiduille testeille toimii sama ajattelutapa kuin muillekin testeille. Ne jaetaan yleensä kolmeen ryhmään: valkoinen, musta- ja harmaalaatikkotestausmalli, jotka tunnetaan paremmin nimillä white box-, black box- ja grey box -testaus. White box -testauksen ajatusmallissa tunnetaan testattava ohjelmisto tai komponentti. Tämän tyyppinen testaus toteutuu yksikkötesteissä ja alustavan toiminnan testauksessa. Black box -testauksessa ei sen sijaan tunneta ohjelmistoa, mutta tiedetään sen ominaisuudet ja miten niitä käytetään. Tämänlainen testi on esimerkiksi loppukäyttäjälle näkyvän käyttöliittymän testaus ja sen toiminnan varmistaminen. (Dustin ym. 2009, 44)



KUVA 5. White-, Grey- ja Black box -testaus. (Madsen, Christiansen & Thordarson 2016)

Kuvassa 5 on näkemys grey box -testauksesta. Grey box -testauksessa käytetään white box-menetelmiä ja black box -testauksen työkaluja. Ohjelmistossa voisi olla virhe, jonka vuoksi tiedot eivät olisikaan tallentuneet tietokantaan, eikä siitä ilmoitettaisi käyttöliittymän kautta. Grey box -testaus on tietojen muuttamista käyttöliittymästä ja muuttuneiden tietojen tallentamisen tarkistamista tietokannasta. Näin voidaan varmistua tietojen tallentumisesta oikeaan paikkaan.

4.2 Kehittämisen syitä ja hyötyjä

Ohjelmistot ovat yhä monimutkaisempia ja sisältävät yhä useampia ominaisuuksia. Tämän vuoksi ohjelmakoodia on paljon ja mahdollisten virheiden määrä kasvaa. Aina ei ole mahdollista tai järkevää testata kaikkea manuaalisesti. Siksi jonkinasteisesta testauksen automatisoinnista on hyötyä. (Dustin ym. 2009, 46–47)

Tuotejulkaisuja ja ominaisuuksia on saatava omaan ohjelmistoon enemmän ja nopeammin kilpailijoihin verrattuna. Ohjelmistojen kehitysaika lyhenee, mutta samalla täytyy saada enemmän valmista koodia. Usein tuotekehitys ei onnistu suunnitellussa aikataulussa, joten jostain on karsittava. Karsittavaksi saattaa valikoitua helposti testaukseen tarvittava aika, ja kun testaus jää vähemmälle huomiolle, tuotetaan entistä epävarmemmin toimivia ohjelmistoja. Tesseyn (2002) raportissa on arvioitu, että ohjelmistojen virheet maksoivat koko Yhdysvaltojen taloudelle noin 59,5 miljardia dollaria vuodessa. Tässä oli laskettu yhteen suorat vaikutukset ohjelmiston kehittäjille ja epäsuorat vaikutukset ohjelmistoja käyttäneille yrityksille. Raportissa selvitettiin vain Yhdysvaltojen tilannetta, mutta voidaan olettaa, että riittämättömän testauksen seurauksena muuallakin maailmassa voidaan havaita suuria taloudellisia menetyksiä. (Tessey, 2002)

Yksi parhaiten perusteltu syy aloittaa testauksen automatisointi on rahan säästö. Perustelu on tärkeä, mutta tärkeitä perusteluja ovat myös ajan säästäminen manuaalisen työn vähenemisen kautta. Yleisesti ohjelmistoyrityksissä tämä tiedostetaan ja testauksen automatisointia pidetään kannattavana. (Dustin ym. 2009, 108)

Ohjelmistojen testaus ihmisvoimin ilman kokemusta ja ohjeistusta on epätasaista ja hidasta. Automatisoitu testi poistaa ihmisen aiheuttaman epävarmuustekijän testauksesta, jolloin testi voidaan toistaa aina samalla tavalla. Jo toteutettu testauksen automatisointi

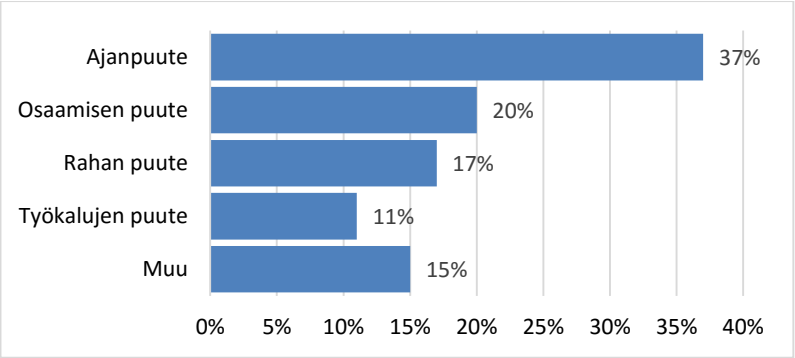
toteuttaa testin aina samalla tavalla, jolloin siihen voidaan luottaa ja mahdollinen virhe on testattavassa ohjelmistossa esiintyvä. (Dustin ym. 2009, 40)

Automatisoidun testauksen toiminnan tavoitteena on hyötyä siitä pitkään ja monipuolisesti. Testien automatisoinnin alussa on päätettävä, mitä testejä ja ohjelmiston osa-alueita halutaan automatisoida. Tulisi pohtia onko joitakin osa-alueita mahdollista tai järkevää automatisoida tai kuinka paljon jonkin testin automatisoinnissa voidaan säästää aikaa tulevaisuudessa. Usein toistettava testi kannattaa automatisoida, jolloin testaajalle jää enemmän aikaa muihin työtehtäviin. Ajan säästäminen on yritykselle automatisoinnin tärkein hyöty. (Dustin ym. 2009, 40)

Automaattisen testauksen hyöty kehittäjille on nopeampi palaute uusista ominaisuuksista. Ilman automatisoituja testejä kehittäjä voi saada palautetta vasta usean päivän kuluttua ominaisuuden implementoinnista tai vasta lopputestauksessa. Jatkuvalla automaattisella testauksella, saadaan nopea palaute kehittäjälle hänen juuri toteuttaneesta ominaisuudesta tai virheen korjauksesta. Huomioitavaa on kuitenkin, ettei automatisointi säästä välittömästi aikaa, vaan voi alkuun vaatia sitä jopa enemmän kuin tavanomainen testaus sen vuoksi, että tarvittavia työkaluja ei välttämättä ole valmiiksi ja niitä täytyy kokeilla. Työkalujen valitsemisen jälkeen niiden täydelliseen hallintaan kuluu vielä lisää aikaa ennen kuin saadaan kaikki hyöty automatisoinnista. (Dustin ym. 2009, 40, 114)

4.3 Toteuttamisen riskejä

Automatisoitu testaus vaikuttaa kannattavalta panostukselta, mutta todellisuudessa se ei välttämättä olekaan, sillä huonosti suunnitellusta tai toteutetusta automaatiosta voi olla enemmän haittaa kuin hyötyä. IDT:n kyselyyn vuonna 2007 automatisoidun testauksen kehittämisestä osallistui yli 700 ohjelmistoalan ammattilaista ja testausinsinööriä pääosin Yhdysvalloista. Kuvassa 6 on esitelty tulokset automatisoinnin epäonnistumisen syistä yrityksissä. (Dustin ym. 2009, 107)



KUVA 6. Automatisoidun testauksen epäonnistumisen syyt. (Dustin ym. 2009, 107)

Kyselyyn vastanneista 72 % kertoo automaation olevan hyödyllistä, mutta automatisointia ei ole toteutettu lainkaan tai vain pienissä määrin vaihtelevin tuloksin. Kuvasta 6 nähdään, että ylivoimaisesti suurin yksittäinen syy automatisoinnin epäonnistumiselle oli ajanpuute. Ajanpuute on monissa projekteissa yleistä vaatimusten kiristymisen takia, joten suurempi aika varataan tuotteen kehittämiselle kuin testaamiselle. Seuraaviksi suurimpina tekijöinä olivat osaamisen, rahan ja tarvittavien työkalujen puute. Erityismaininnan automatisoinnin ongelmista kyselyssä saivat automatisointiin liittyvät ennakkokäsitykset, automatisointiprosessien kuvausten puutteellisuus tai puuttuminen, ohjelmistokehityksen aliarviointi testauksen automatisoinnissa ja yleisten standardien puute. (Dustin ym. 2009, 107–108)

Toiseksi suurimpana syynä pidettiin osaamisen puutetta. Testauksen automatisoinnin toteutus varsinkin sen kehityksen alkuvaiheessa voi vaatia tuotekehitykseltä apua. Ohjelmistokehittäjät tekevät usein vain yksikkötestejä kehittäessään ohjelmistoa ja testaavat tekemänsä, mutta aina tarvitaan laajempaa, koko tuotteen kattavaa testausta, kun yksittäisiä komponentteja liitetään toisiinsa. Automatisoinnin toteutuksen jäädessä vain testaajan taitojen varaan siitä ei välttämättä tule alkuperäisen tarkoituksen mukainen. (Dustin ym. 2009, 111)

Neljänneksi suurimpana ongelmana pidettiin automatisoinnin työkalujen puuttumista. Tätä voidaan pitää ongelmana yrityksissä, joissa kehitetään samaa tuotetta monelle eri alustalle. Ohjelmistoa kehitettäessä Windows-, Linux- ja Mac-laitteille olisi kannattavaa toteuttaa automatisointi yhdellä työkalulla. Automatisoinnissa on valmistauduttava siihen, ettei yksi työkalu riitä vaan tarvitaan useampaa työkalua. (Dustin ym. 2009, 113–115)

Testauksen automatisoinnin kehittämisen aloittamisen sanotaan olevan kallista ja epävarmaa. Toteuttajan kokemus ja osaaminen ovat tärkeitä seikkoja onnistumisen kannalta, mutta eivät välttämättömiä. Mahdollisesti suurin kynnys automatisoinnin aloittamiselle on sen pitkäaikainen kehittäminen. Se sitoo henkilöstön aikaa testien tekemiseen, mikä on pois muusta työnteosta. Pahimmassa tapauksessa voidaan tehdä tarpeettomia testejä, jotka eivät toimi uusien ominaisuuksien lisäämisen jälkeen. Hyvällä suunnittelulla voidaan hallita ongelmia ja riskejä. (Dustin ym. 2009, 114)

4.4 Testiautomaatiotyökaluja

Automaatiotyökaluja on tehty eri tarkoituksiin. Monet niistä ovat maksullisia, mutta jotkut toimivat Open source -lisenssillä. Joiltakin yrityksiltä on saatavilla myös ilmainen työkalu, johon on myynnissä lisätyökaluja ja koulutusta. Avoimeen lähdekoodiin perustuvia työkaluja voidaan yhdistellä vapaasti muiden avoimen lähdekoodin työkalujen kanssa. IT-alan yritykset voivat tarjota palveluna testausautomaation kokonaisvaltaista rakentamista. Tämän kaltainen palvelu auttaa, kun tarvitaan osaavaa työvoimaa automaation kehittämiseen. (Dustin ym. 2009, 40)

Jotta testausautomaatio olisi kannattavaa, täytyy työkaluja vertailla huolellisesti. Tutkittaessa työkaluja yksi huomionarvoisimmista asioista on työkalussa tuetut käyttöjärjestelmät. Ei ole mitään järkeä tuhlata aikaa työkaluun, joka toimii vain Windows-käyttöjärjestelmällä, jos tulevaisuudessa on tarkoitus testata vaikkapa Windows:n lisäksi iOS- ja web-pohjaista versiota tuotteesta. (Dustin ym. 2009, 37)

Toinen tärkeä asia työkalun valinnassa on, mitä ohjelmointikieliä se tukee eli millä ohjelmointikielellä testattava ohjelma toteutetaan. Kannattaa miettiä testattavan ohjelmiston tulevaisuutta, jotta säästytään kalliilta ja aikaa vievältä testausohjelman vaihdolta, kuten esimerkiksi päätetään vaihtaa ohjelmointikieli toiseen, kuten Java johonkin muuhun ohjelmointikieleen. Silloin työkalun kannattaa tukea uutta kieltä, jotta ohjelmointikielen muuttuminen ei vaikuttaisi suuresti testauksen automatisointiin. (Dustin ym. 2009, 35–37)

Testausautomaatioon tarkoitettuja työkaluja on runsaasti ja paras työkalu riippuu tarpeesta. Kun halutaan testata web-pohjaista ohjelmaa, valitaan työkaluksi luultavasti erilainen kuin silloin, jos testataan verkkoliikennettä tai älypuhelinsovellusta. Joskus täysin valmista työkalua ei ole saatavilla valmiina, mutta tällöin on hyödyllistä tutkia, olisiko jokin valmis ohjelma hyvä pohja oman työkalun kehittämiseksi. (Dustin ym. 2009, 61)

Myös sellainen työkalu on tarpeen, joka mahdollistaa komentojen ajamisen automaattisesti. Tällaisia työkaluja kutsutaan englannin kielellä nimellä continuous integration- eli CI-työkalut, mikä tarkoittaa vapaasti suomennettuna yhtäjaksoista integrointia. Näitä työkaluja on esimerkiksi Jenkins, Buildbot, Travis CI ja TestRail, jotka muodostavat yhteyden toisiin koneisiin, joissa ne suorittavat CI-koneille ennalta määritettyjä komentoja. CI-toimintoja voidaan ajastaa kellon tai tapahtumien mukaan. Usein nämä koneet rakentavat eli buildaavat kehitettäviä ohjelmistoja ja toteuttavat niille yksikkötestejä. (Bedell 2011)

5 TESTAUKSEN AUTOMATISOINNIN MAHDOLLISUUKSIA REITITTIMELLE

5.1 Reitittimen testauksesta

Reitittimen valmistajasta riippuen sen käyttö ja annettavat komennot saattavat poiketa toisistaan. Reitittimien valmistajia on useita, ja näistä tunnetuin lienee Cisco. Muita tunnettuja valmistajia ovat Zyxel, D-Link, HP, Juniper ja Alcatel-Lucent. Kaikkien valmistajien laitteiden käyttäminen on toteutettu eri tavoilla, mutta protokollat ovat standardisoituja ja toimivat samalla tavalla. Testauksen automatisoinnin kannalta laitteella ei ole väliä, koska komentoja voidaan muuttaa tarpeen mukaan.

Suunniteltaessa testauksen automatisointia on hyvä selvittää olemassa olevien testaustyökalujen hyödyntämismahdollisuuksia. Toteutus kannattaa suunnitella siten, että ohjelmiston muuttuessa ei tarvitse tehdä suuria muutoksia automaatioon.

Testauksen automatisointi voidaan toteuttaa reitittimelle samalla tavalla ja samoilla työkaluilla kuin käsin testattaessa. Seuraavissa luvuissa tutustutaan joihinkin testausmenetelmiin, joita voidaan automatisoida. (Dustin ym. 2009, 317)

5.1.1 Pakettianalysointi

Pakettianalysointitekniikalla voidaan tutkia verkossa kulkevia viestejä ja niiden otsikkokenttiä. Tällä menetelmällä voidaan tarkkailla verkossa kulkevia paketteja, kuten UDP- tai multicast-viestejä. Pakettien otsikkokentän avulla voidaan myös varmistua reititysprotokollista sekä esimerkiksi varmistua IKE-neuvotteluiden tapahtumisesta. Pakettianalysointi mahdollistaa pakettien salauksen varmistamisen. (Parziale ym. 2006, 829)

Jotta otsikkokenttiä voisi tulkita, täytyy olla mahdollisuus kaapata liikennettä. Liikennekaappaustyökaluilla, kuten wireshark ja tcpdump, voidaan tutkia verkkoliikennettä. Niillä voi havaita, minkälaisia paketteja verkossa liikkuu. Protokollat ovat standardoituja, joten otsikkokentän tiedoista voidaan päätellä, mikä protokolla on kyseessä. Jokaisessa paketissa nähdään myös vastaanottajan ja lähettäjän IP-osoitteet. (TCPDUMP, 2015)

Pakettianalysoinnilla voidaan tutkia verkossa kulkevaa liikennettä ilman pääsyä reitittimelle. Vaatimuksena pakettien analysoinnille on pääsy samaan verkkoon, jota halutaan tutkia ja työkalu pakettien analysointia varten. (TCPDUMP, 2015)

5.1.2 Ping ja liikenteen lähettäminen

Liikenteen lähettäminen verkosta toiseen on yksinkertainen tapa varmistaa, että reititin välittää paketteja. Usein testataan ensin liikenteen kulkeminen verkosta toiseen ja vasta sen jälkeen tarkistetaan muut toiminnot. Ping on ICMP-viestiä lähettävä työkalu, jolla voidaan varmistaa pakettien kulkeminen verkosta toiseen. (Internet Control Message Protocol, 2016)

Verkossa liikkuu ICMP-viestien lisäksi myös esimerkiksi TCP- tai UDP-viestejä. TCP:tä käytetään, kun halutaan varmistua pakettien eheydestä ja saapumisesta vastaanottajalle. UDP:tä taas hyödynnetään, kun paketit täytyy saada nopeasti lähettäjältä vastaanottajalle, eikä yhden paketin hävittämisellä ole suurta vaikutusta palvelun toimintaan. (Internet Control Message Protocol, 2016)

Reititin voi toimia myös palomuurina. Tällöin pelkkä ping-työkalulla tarkistaminen ei riitä, vaan on käytettävä jotain muutakin työkalua. Monipuolisempi työkalu liikenteen lähettämiseen on netcat, jolla voidaan lähettää TCP- tai UDP-paketteja ja määritellä käytettävä porttinumero. Tällöin voidaan tarkistaa reitittimen porttien rajaustoimintaa yhdellä työkalulla. (Netcat 1.10, 2016)

5.1.3 Asetusten tarkistaminen

Reitittimen asetuksiin päästään usein käsiksi SSH-, http- tai sarjakaapeliyhteydellä. Ciscon reitittimelle voidaan antaa komento show running-configuration, jolloin se tulostaa näytölle sillä hetkellä voimassa olevat asetukset. Tulostuksesta voidaan tarkistaa porttien IP-osoitteet tai reititysprotokollan toiminta. Vaikka asetukset näyttäisivät oikeilta reitittimellä, niihin ei voi aina luottaa. Sen vuoksi on käytettävä muitakin tapoja reitittimen testauksessa. (Basic Router Configuration, 2016)

Reititys on reitittimen tärkein tehtävä, joten on hyvä tarkistaa, että reitit näkyvät oikein. Linux-koneella tämä tarkistetaan ip route-komennolla. Tulostetusta taulukosta nähdään koneen reitit muihin verkkoihin ja näin varmistutaan, että reititys ja tietoliikenneverkko toimivat halutulla tavalla. (Brown, 2007)

5.2 Automatisointityökaluja

5.2.1 Jenkins

Luvussa 4.4 kerrottiin, että testausautomaatio tarvitsee jonkin työkalun, jolla automatisoidaan testit. Valmiita työkaluja on saatavilla, mutta usein niitä täytyy muokata, jotta ne sopivat omaan käyttötarkoitukseen. Testausta toteuttavan skriptin lisäksi tarvitaan siis jokin ohjelma, joka automatisoi testin. Muuten varsinaista testauksen automatisointia ei ole tapahtunut, vaan on luotu pelkästään työkaluja helpottamaan manuaalista testausta.

Jenkins on yksi työkalu, jolla voidaan automatisoida testaus. Se on tarkoitettu automatisoimaan usein toistettavat tehtävät ohjelmistokehityksessä. Jenkins-ohjelmalle määritetään koneita, joihin se voi olla yhteydessä ja koneille annetaan tehtävä, jossa kerrotaan, mitä koneen täytyy suorittaa. Tällä tyylillä voidaan automatisoida valmiiden testiskriptien ajamista, jolloin erillistä testaustyökalua ei tarvita. (Kawaquchi, 2016)

Jenkins ei ole riippuvainen muista ohjelmista, mutta monet ohjelmat tukevat sitä. Siihen on myös liitännäisohjelmia, joilla voidaan laajentaa sen käyttömahdollisuuksia eri tarkoituksiin. Liitännäisten avulla voidaan saada esimerkiksi raportteja tehtävistä tai testien tuloksista. Jenkins tai jokin muu CI-työkalu on tarpeellinen ja usein myös välttämätön, kun kehitetään testauksen automatisointia. (Kawaquchi 2016)

5.2.2 CDRouter

CDRouter on reitittimen automaattinen testaustyökalu. Sitä on helppo käyttää ja se sisältää jo valmiita testejä. Perusversio on maksullinen ja sisältää yli 600 valmista verkkotestiä. CDRouter-työkalu ei tue virtualisointia, vaan se tarvitsee oman tietokoneensa tai tarkoitukseen kehitetyn laitteen toimiakseen. Käyttäjän tehtävänä on kirjata testattavan reitittimen asetukset, joita hyödynnetään ohjelman testeissä, minkä jälkeen valitaan ohjelman testeistä tarvittavat. Jos sopivaa testiä ei löydy, voi työkaluun kirjoittaa omia testiskriptejä. Testejä voi toistaa jatkuvasti, ajoitetusti tai vain kerran, ja jokaisesta testistä ohjelma luo oman raporttinsa. Jos testeissä lähetetään verkkoliikennettä, kaikki verkossa kulkeva liikenne kaapataan ja tallennetaan raportteihin. (CDRouter Quick Start Guide, 2016)

CDRouter-työkalu on kehitetty yhden reitittimen toiminnan varmistamiseksi, joten sillä ei voida testata useampaa reititintä kerrallaan. Sen käyttöönotto on helppoa ja nopeaa, joten hyödyn ohjelmasta saa nopeasti. Kaikkia valmiita testejä voidaan muokata tarpeen vaatiessa. (CDRouter Quick Start Guide, 2016)

5.2.3 EasyTest

EasyTest on Alacatel-Lucentin kehittämä työkalu verkkolaitteiden testaamiseen. Se on julkaistu vapaana lähdekoodina vuonna 2012 ja siitä on versiot Windowsille ja Linuxille, mutta vain Windows-versio on saatavilla vapaasti. EasyTest on tehty yksinkertaiseksi, joten ei tarvita kokemusta ohjelmoinnista.

EasyTest tukee monia yhteysprotokollia, kuten telnet, SSH, FTP ja TCP-socket. Siihen sisältyy kattavat ohjeet testien tekemisestä ja miten niitä voidaan muokata jälkikäteen. Testit tehdään nauhoita ja toista -periaattella, mutta työkalu tukee myös omien skriptien käyttöä. EasyTest-työkalussa voidaan määritellä testien suoritusajankohta, minkä lisäksi ohjelman voi liittää Jenkins-työkaluun. EasyTestissä on myös valmis testiraporttiominaisuus. Koska kyseessä on avoimen lähdekoodin ohjelma, tukea työkalulle ei ole saatavilla virallisesti. (Zhong Z, 2013)

Työkalun käytön alussa määritellään tiedot testausympäristöstä, esimerkiksi mitä laitteita tullaan käyttämään testauksessa. Tämän jälkeen tehdään testi, jonka työkalu nauhoittaa ja erittelee jokaisen yksittäisen toimenpiteen omaksi askeleekseen. Näihin askeliin voidaan lisätä toimintoja, kuten tarkistuspisteitä tai muita vertailuja. Näin voidaan toteuttaa luotettava testausautomaatio. (Zhong Z, 2013)

5.2.4 Twister

Twister on Luxoftin kehittämä testausautomaatiotyökalu verkkolaitteille. Luxoft on erikoistunut muun muassa tietoverkkojen kehittämiseen, rakentamiseen ja testaamiseen. Twister on avoimeen lähdekoodiin perustuva ohjelma, joka vaatii kokemusta ohjelmoinnista. Sillä voidaan testata kaikkia verkossa olevia laitteita samanaikaisesti, mikä mahdollistaa laajan ja monipuolisen testausautomaatiokokonaisuuden rakentamisen. Twister tukee kolmea skriptauskieltä, jotka ovat TCL, Python ja Perl. Työkaluun on saatavilla monia lisäosia helpottamaan testausautomaation rakentamista, kuten versionhallinta- tai verkkoliikenteen kaappaustyökaluja. Lisäksi raportointityökalulla saadaan yksityiskohtaisia ja laajoja yhteenvetoja kaikista ajetuista testeistä. (Twister: Product, 2013)

Twisterin ohjeet ovat kattavat ja virallista tukea on saatavilla, vaikka kyseessä on vapaaseen lähdekoodiin perustuva työkalu. Työkalua on vaikeaa käyttää ilman kokemusta ohjelmoinnista, koska Twisterissä ei ole testien tekoon vastaavia valmiita työkaluja kuin aiemmin esitetyissä ohjelmissa, mutta se on vakaa alusta testauksen automatisoinnin kehittämiselle. (Twister: Product, 2013)

6 POHDINTA

Työssä tarkasteltiin useampaa testauksen automatisointiin kehitettyä työkalua. Jenkins tai jokin muu CI-työkalu on välttämätön automatisoitaessa testausta. Sellaisella voidaan ajoittaa automaattiset testit haluttuun kellonaikaan tai tapahtumaan. CDRouter, EasyTest ja Twister on kehitetty verkkotestauksen automatisointiin. Jokaisella näistä on oma näkökulmansa reitittimien testaukseen, joten vain yhtä parasta vaihtoehtoa ei ole. Kaikissa ohjelmissa on testausraportointityökalut, jolloin automatisoinnista saadaan todellista hyötyä virheen sattuessa.

CDRouter on hyvä vaihtoehto, jos halutaan testata ja varmistaa vain yhden reitittimen toiminta. Se sopii varsinkin silloin, kun halutaan testata uuden laitteiston ja ohjelmiston toimintaa yhdessä. Se sopii tavallisen kotireitittimen tai monimutkaisemman yritystason reitittimen testaukseen. Etenkin pienelle kehitystiimille CDRouter voi tuoda huomattavaa apua, koska perustoiminnallisuudet voidaan varmistaa helposti. Hyvä puoli CDRouterissa on siihen sisältyvät valmiit testit, jolloin testauksen automatisoinnista saadaan nopeasti hyötyä. Ohjelman huono puoli on se, ettei useita reitittimiä voida testata yhtä aikaa eikä erilaisia verkkoympäristöjä voida rakentaa.

EasyTest on CDRouteria monipuolisempi testaustyökalu. Se on myös kokemattomalle ohjelmoijalle hyvä vaihtoehto, koska sillä testaaminen ei vaadi ohjelmointikokemusta. Testien nauhoittamistoiminto on hyvä ominaisuus, jolla voidaan tehdä nopeasti testejä. Ohjelmasta on saatavilla versio vain Windowsille, mutta ei Linuxille. Työkalulle ei ole kunnon tukea eikä sitä ole päivitetty vuosiin. Toisaalta ohjelmassa on hyvä runko, jota voidaan jatkokehittää testauksen automatisointiin.

Twister on näistä kolmesta monimutkaisin ja vaikeasti lähestyttävin. Se ei sisällä valmista pohjaa testien tekoon, vaan sillä on tarkoitus ajaa testaukseen toteuttavia skirptejä. Jos on valmiita testiskriptejä, niin niitä voidaan helposti hyödyntää ohjelmassa. Osaavissa käsissä Twister on hyvä alusta, jolla voidaan toteuttaa reitittimen automatisoitu testaus. Ilman ohjelmointitaustaa testaajan on vaikea toteuttaa mitään toimivaa.

Tehokkuus on tärkeää ohjelmistokehityksessä. Yksi tehostuskeino on automatisoida kaikki mahdollinen, esimerkiksi testaus. Ilman kokemusta testauksen automatisointi on

vaikeaa ja aikaavievää. Monesti automatisoinnin toteuttaminen on pitkäaikainen prosessi, joka ei välttämättä lopu koskaan. Tämä ei kuitenkaan merkitse testauksen automatisoinnin epäonnistumista, vaan ohjelmistoa kehitettäessä testausautomaatiota on kehitettävä. Aina ei ole edes järkevää yrittää automatisoida kaikkia harvoin toistettavia asioita. Testauksen automatisoinnista on hyötyä pitkällä aikavälillä, vaikkei alussa siltä tuntuisikaan. Tarvitaan joka tapauksessa aikaa ja resursseja. Automatisoidulla testauksella voidaan poistaa yksinkertaisten ja helppojen testien käsin tekeminen, jolloin resursseja voidaan kohdistaa monimutkaisiin ja vaikeisiin testeihin. Tämä hyödyttää koko ohjelmistokehitystä ja mahdollistaa laadukkaampien tuotteiden synnyn.

LÄHTEET

Basic Router Configuration. 2016. Cisco. Luettu 2.4.2016
http://www.cisco.com/c/en/us/td/docs/routers/access/1800/1801/software/configuration/guide/scg/routconf.html#15062

Bedell C. Continuous integration: Tools and trends. 2011. TechTarget. Luettu 5.4.2016
http://searchsoftwarequality.techtarget.com/feature/Continuous-integration-Tools-and-trends

Brown M. 2007. Ip route. Luettu 19.4.2016
http://linux-ip.net/html/tools-ip-route.html

CDRouter Quick Start Guide. 2016. QaCafe. Luettu 18.4.2016
http://support.qacafe.com/quick-start-guide/cdrouter-quick-start-guide/

Colliander A. 1999. ISOn OSI-mallin rakenne ja käyttö. Luettu 5.3.2016
http://www.tml.tkk.fi/Studies/Tik-110.300/1999/Essays/essee_OSI.html

Configuring DHCP. 2001, päivitetty 2006. Cisco. Luettu 2.4.2016
http://www.cisco.com/c/en/us/td/docs/ios/12_2/ip/configuration/guide/fipr_c/1cfdhcp.html

Configuring IPSec Network Security. 2016. Cisco. Luettu 2.4.2016
http://www.cisco.com/c/en/us/td/docs/security/vpn_modules/6342/vpn_cg/6342site3.html

Configuring Network Address Translation: Getting Started. 2014. Cisco. Luettu 2.4.2016
http://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/13772-12.html

Configuring a Simple Firewall. 2016. Cisco. Luettu 2.4.2016
http://www.cisco.com/c/en/us/td/docs/routers/access/1800/1801/software/configuration/guide/scg/firewall.html

Configuring VLANs. 2016. Cisco. Luettu 2.4.2016
http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli/CLIConfigurationGuide/VLANs.html

Doyl J., Carrol J. Routting TCP/IP, Volume 1. 2005. Flylib. Luettu 1.4.2016
http://flylib.com/books/en/2.296.1.21/1/

Dustin, E., Garret, T. & Gauf, B. 2009. Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality. Elfriede Dustin, Thom Garrett, Bernie Gauf. Yhdysvallat: Pearson Education.

Internet Control Message Protocol. 2016. Wikipedia. Luettu 2.4.2016
https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

Kasurinen, J. P. 2014, Ohjelmistotestauksen käsikirja. Suomi: Docento.

Kawaquchi K. Use Jenkins. 2016. Jenkins. Luettu 16.4.2016
https://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins

Madsen, Christiansen & Thordarson. Grey Box. 2016. DTU Compute. Department of Applied Mathematics and Computer Science. Luettu 26.2.2016
http://energy.imm.dtu.dk/models/grey-box.html

Malaiya. Automatic Test Sofware. 1999. Computer Science Department. Colorado State University. Luettu 25.2.2016
http://www.cs.colostate.edu/~malaiya/tools2.pdf

Netcat 1.10. 2016. Sourceforge. 2.4.2016
http://nc110.sourceforge.net/

Parziale, L., Liu, W., Matthews, C., Rosselot, N., Davis, C., Forrester, J., Britt, D. T. 2006. TCP/IP Tutorial and Technical Overview. Yhdysvallat: IBM Redbooks

Router. 2016. Computer Hope. Luettu 2.4.2016
http://www.computerhope.com/jargon/r/router.htm

Tassey. 2002. Economic Impact of Inadequate Infrastructure for Software Testing. NIST. Luettu 25.2.2016
http://www.nist.gov/director/planning/upload/report02-3.pdf

TCPDUMP. 2015. tcpdump. Luettu 2.4.2016
http://www.tcpdump.org/tcpdump_man.html

Troubleshooting, Fault Management, and Logging. 2001 päivitetty 2004. Cisco. Luettu 2.4.2016
http://www.cisco.com/c/en/us/td/docs/ios/12_2/configfun/configuration/guide/ffun_c/fcf013.html

Twister: Product. 2013. Luxoft. Luettu 19.4.2016
http://twistertesting.luxoft.com/product/

Zhong Z. Test Made Simple - EasyTest Automation Test Tool. 2013.Alcatel-Lucent. Luettu 18.4.2016